Study of Web Application Attacks & Their Countermeasures

Naresh Kumar[1], Kanika Sharma[2]
[1, 2] University Institute of Engineering & Technology, Kurukshetra University, India
[1]Naresh_duhan@rediffmail.com
[2] Kanikasharma1312@gmail.com

***Abstract*— Web application security is among the hottest issue in present web scenario due to increasing use of web applications for e-business environment. Web application has become the easiest way to provide wide range of services to users. Due to transfer of confidential data during these services web application are more vulnerable to attacks. Web application attack occurs because of lack of security awareness and poor programming skills. According to Imperva web application attack report [1] websites are probe once every two minutes and this has been increased to ten attacks per second in year 2012. In this paper we have presented most common and dangerous web application attacks and their countermeasures.**

***Index Terms*— Security, Web Application Attacks, SQL Injection, Cross Site Scripting, XSS, Broken Authentication, Session Management.**

I. Introduction

Web applications are increasingly becoming an easy way for exchanging information over internet. They simply provide an interface through which client can communicate. For handling client information, service providers uses database which stores client sensitive data. This database is exploited in many ways by the attackers for gaining users personnel information. Web applications are the easy way for attackers to access underlying database as they are often vulnerable to attacks. According to open web application security project (OWASP) among top ten web application vulnerabilities , code injection attack is the most common and most dangerous attack and is at number one position from past seven years followed by cross site scripting and broken authentication and session management. Web application take input from the user through textboxes in the form of name , address , comment feedback and many the ways. These input are connected through the database to store the user input values in the database table. Malicious user injects the SQL query or script to perform injection attacks. These queries are executed by web browser as the code which acts differently as intended by the programmer. Authentication and session attack occurs when malicious user hijacks the ongoing session or bypasses the authentication mechanism. These attacks can provide the privilege of an authenticated user to the malicious user which can cause severe consequences such as loss of confidentiality, integrity, authentication, authorization [20].

As per OWASP 2013 there are ten major categories defined for web application attack in 2013 release [2]

- A1-Injection
- A2-Broken authentication and Session management
- A3-Cross Site Scripting (XSS)
- A4-Insecure Direct Object References
- A5-Security Misconfiguration
- A6-Sensitive Data Exposure
- A7-Missing Function Level Access Control
- A8-Cross Site Request Forgery (CSRF)
- A9-Using Components with Known Vulnerabilities
- A10-Unvalidated Redirects and Forwards.

In real world, it is very difficult to achieve complete security as some security flaws always exist which can attack the application in different ways. In next section we have discuss impact of top three web application attacks SQL injection , XSS , Broken authentication and Session management.

II. SQL Injection

SQL injection (SQLi) attack occurs when malicious user injects SQL keywords as the part of input. SQLi executes due to improper or insufficient validation of input. Through SQLi attacker can gain unauthorized privilege for database, execute commands perform data manipulation operation on database. These queries can dynamically construct as the part of user. Injection attack can be classified in to three categories. *First Order Attack*, in which unions or sub SQL query is injected to the existing query statements. *Second Order Attacks*, in this malicious code is permanently stored in the database. They attack internal application users, system administrators through search list, submissions, latest popular article. *Lateral Injection*, in lateral injection malicious user can attack PL/ SQL procedure that even does not take user input. It occurs when a variable whose data type or number is concatenated with text of SQL statement, there is a chance of injection attack. When an input source is found by the attacker various SQLI types are used to perform attack of different kinds [15].

A. Tautology

A tautology is a statement that is always true. This type of attack is used to bypass the authentication mechanism by using relational operators for comparing operands to generate a condition which is always true. For example

Select * from users where username = '' OR 1=1—' AND password = 'anything';

DOI: 03.LSCS.2013.3.98

A user is the database and username and password are the input fields through which attack is performed. In this ''OR 1=1—"will always return true condition and commonly known as tautology.

B. Piggy-Backed Queries

In this type of attack an additional query is appended to the original query and query is executed as the part of initial query. For example

Select contact from userinfo where login='xyz' AND pinnum ='123'; drop table userinfo;

Here database executes both the queries which drops table. Using this type of attack tables can be added deleted from database.

C. Logically Incorrect Queries

The main intention of this type of attack is to find useful information of the database by generating error message. Such as number of columns of the database , database name ,database version , name and type of each column. This information can be used further for exploiting database.

D. Union Query

Union query injection is also known as Statement injection attack. This attack is performed by inserting union query so that database returns dataset which is union of initial query. For example

Select * from users where username ='' UNION select * from admin—'and password ='anything'"

Here query becomes union of two select queries. First query returns null and second returns all data from table admin.

E. Stored Procedure

Stored procedures are the code and they are vulnerable as program code. For unauthorized or authorize user they return true or false. If the attacker inputs ",;SHUTDOWN;— " for username then stored procedure generates following query

Select login from users where username='';SHUTDOWN;— and password='anything'.

F. Inference

This type of attack is performed to gain information about the vulnerable parameter of the database. This type of attack creates queries so that database or application behaves differently from as intended by the programmer. The two techniques used in this type of attack are: Blind Injection and Timing attacks. In blind injection attacker perform query that have result of true or false. If answer is false then error is generated else application behaves correctly. In timing attack attacker executes query in form of if then statement and uses WAITFOR keyword which causes database to delay its response.

III. Cross Site Scripting

Cross site scripting (XSS) is a common vulnerability in web application programs.XSS occurs when HTML or JavaScript code is injected into the database through input sources. If these inputs are not filtered from server side, then severe consequences may occur such as accessing and transmitting cookies, Redirecting to third party websites. The victim of the XSS are the most common sources such as comments, feedback, search engines. They target authenticated users and system administrators.XSS are classified in two type of attacks.[17].

A. Stored Or Persistent Attack

In stored attacks the malicious code injected by the attacker is stored permanently in the database. The code is accepted as a part of input field. The victim then receive the malicious code when it request the stored information.

B. Reflected Or Non Persistent Attack

Reflected attack occurs off the web server, such as it is present in an error message or in a search result response that includes part of input or all of the input sent to the server as part of the query. They are also delivered to victims via redirects & forwards, such as in an e-mail message, which reflects the user to another link or to a un-trusted source or server.

IV. Broken Authentication & Session Management

Broken authentication and session management attack occurs whenever there is session hijacking or false authentication occurs. It includes management of handling all aspects of authentication and sessions which can affect the web servers, application servers, web application environment and can cause misuse of privileges. For e.g. the attacker can change message or can misuse information retrieved. Various cryptographic algorithms and session management tokens are used by developers for this kind of attacks but still it is a major issue [3].For handling authentication and session issues various issues should be kept in mind such as :

A. Password Strength

Passwords should have minimum length and appropriate use of alphabets, numbers and special keywords to avoid guessing.

B. Password Use

Number of login attempts should be restricted as authenticated users will never re attempts if he forgets the password. After attempted maximum attempt user should be restricted to avoid attack.

C. Password Change Control

A single password change mechanism should be there to avoid the security flaws. The mechanism should ask for old and new password but changing email address or changing phone number should be avoided.

D. Password Storage

Password should be stored in with encrypted or hashed method to protect them from exposure.

DOI: 03.LSCS.2013.3.98

E. Session Id Protection

Session IDs should be long, complicated, having random numbers that cannot be easily guessed. It should be changed frequently during an ongoing session to reduce session ID validity. Session IDs must be changed when performing operations such as switching to SSL, authenticating user, or other major transitions. Session IDs chosen by a visitor should never be accepted as the ID.

V. SQL Injection Detection & Prevention

When an input source is found by the attacker which can be used to perform attack various kinds of techniques can be used to detect and prevent [15].

A. ¡°SQL DOM: Compile Time Checking of Dynamic SQL Statements" (Russell A. McClure and Ingolf H.Kruger) (2005)

Russell proposed an approach [4] in which a set of classes that are strongly type to a database schema is presented. These are the classes which are used to generate SQL statements. The solution consists of an executable, sqldomgen which is executed against database. The output provided by the sqldomgen is a dynamic link library (DLL) which contains classes that are strongly typed to database schema. These classes are known as SQLDOM through which application developer constructs dynamic SQL statements .If database schema changes, sqldomgen is rerun generating modified SQLDOM.SQLDOM consist of three parts. First is abstract object model which is used to construct every possible valid SQL statement which would execute at runtime. Second is SQLDOM generator for dynamically generating code. Third is concrete object model having three main types of classes SQL statements, columns, where conditions. The weaknesses of this approach are the time needed to construct SQL statements is more and complexity may occur with large databases as sqldomgen need to be updated frequently.

B. "Static Analysis Framework For Detecting SQL Injection Vulnerabilities" (Xiang Fu ,Xin Lu) (2007)

Xiang Fu [5] proposes the static analysis framework for identifying SQL injection vulnerabilities at compile time. This framework consists of MSIL Instrumentor which instruments byte code of an ASP.NET application. A Symbolic Execution Engine which examines back end code of each ASP page. A Rule Library having attack patterns. A Constraint Solver which generates valuation of variables that satisfy constraint. A Test Case Generator which carries test for the vulnerability of the program. At each input box which accepts SQL query, a hybrid constraint solver is applied to find out the user input that can be malicious and can lead to attack. The weakness of this static analysis technique is that it has not been implemented yet and designed especially for the .NET framework. After implementation of this framework it has potential to discover more attacks than black box web security testing tools.

C. "Web Application Intrusion Detection System for Input Validation Attack (WAIDS). "(YongJoon Park , JaeChul Park) (2008)

YongJoon Park proposed [7] an approach based on web application parameters which has identical structures and values. WAIDS is an intrusion detection method based on Anomaly Intrusion Detection model for detecting input validation attack against web applications. This approach has four steps. In step one parameters in HTTP request is collected. In second step the collected data keyword is transformed in to alphabetic characters for data filtering according to keyword replacement matrix. In third step web application request profile similarity is measured to find most appropriate normal web request. For this keyProDiff formula is used which is $OSD(P,S) = \min \{OSD_{keyrodiff}(P,S)\}$.In this step at runtime HTTP request are checked against normal web request profile. Malicious code is detected and reported those that violate normal profile. This approach has three modules. First is data collection module which collects data. Second is Analysis module which implements step two and three. Third is management module which manages attack detection and report logging. The weakness of this approach is that the profile matching is very exhaustive task and developer knowledge is required.

D. "An Automatic Mechanism For Sanitizing Malicious Injection " (Jin Chernf Lin , Jan-Min Chaen ,Cheng-Hsiung Liu) (2008)

Jin Cherng Lin has proposed [8] an automatic mechanism for sanitizing malicious injection. It evaluates the filter rules for detection rate to prevent web application attacks. They suggested a hybrid analysis methodology in which they performed at three levels: by a white list, blacklist and encoding technique. The proposed testing framework test different cases. If the output of tester and the monitor which accept input matches then there is no attack detected. The drawbacks of this testing framework are the exhaustive test cases and the occurrence of false positive.

E. "SQLProb : A Proxy based Architecture towards preventing SQL injection attacks " (Anyi liu ,yi yuan) (2009)

Anyi Liu proposed an approach [9] this approach has four main components. First is Query Collector which processes all possible SQL queries during data collection phase. Second is User Input Extractor to identify user input data. Third is Parse Tree Generator for generating parse tree for input queries. Fourth is User Input Validator evaluates whether user input is malicious or not. SQLProb executes in two phases the first is Data Collection Phase which collects the user input data. Second is Query Evaluation Phase which evaluates the malicious query. This framework is implemented in java and tested on virtual machine with MYSQL as backend database server. The advantage of this approach is that it does not require access to source code of web application and it is easily deployable to existing enterprise environment.

DOI: 03.LSCS.2013.3.98

F. "An approach For SQL injection Vulnerability detection"(MeiJunjin) (2009)

MeiJunjin proposed an approach [10] for detecting SQL vulnerability. This approach traces the flow of input values that are used for SQL query by using AMNESIA SQL query model. Based on the input flow analysis, he generates test attack input for the method arguments. used to construct SQL query. This approach also generates a colored call graph indicating secure and vulnerable method. To calculate efficiency the case studies were performed on two web application. The advantage of this approach is that it does not have any false positive. The weaknesses of this approach are that false negative is there. It is tested for small web application so false positive and negative rate may get varies.

G. "Multi-Layered Defense against Web Application Attacks " (Abdul Razzaq ,Ali Hur , Nasir Haider ,Farooq Ahmed) (2009)

In [11] Abdul Razzaq proposed an approach of multilayer defenses to the application level attacks. In this approach there are two modules. First is, Detection Module in which special characters are recognized through three components Positive Security, Negative Security and Anomaly Detection. If keyword is matched in any of component further processing is stopped. If there is no vulnerability detected then input is passed to next module. Analyzer and Validation is second module which generates exception for error. The weaknesses of this approach are, it needs developer knowledge. False positive are also detected in this approach.

H. "A Database security testing scheme of web application" (Yang Haixia and Nan Zhihong) (2009)

Yang Haixia and Nan Zhihong proposed [12] database security testing scheme of web application. In this approach they suggested a testing model for securing database. They develop attack rule library having various injection patterns. Whole website is scanned to find software faults. The test reports are generated for the test cases. This method automatically detects input points of SQL attacks. The drawback of this framework is that it can detect only signature SQL injection attack.

I. "Injection Attack Detection using the removal of SQL Query Attribute values " (Jeom-Goo Kim) (2011)

Jeom-Goo Kim proposes [13] a simple effective SQL Query removal method which uses combined technique of static and dynamic analysis. This method compares and analyzes input by removing the attribute value of SQL queries. A function F is proposed which has functionality to delete the attribute value in SQL queries. The attribute value of static SQL query in web application and SQL queries generated by the runtime will be deleted. If after removing the attribute the query matches the fixed SQL query then there is no attack detected. But if some difference is there in query then attack is considered. The weakness of this application is that it has not been implemented yet.

J. "Research of intelligent intrusion detection system based on web data mining technology" (Chai Wenguuang, Tan Chunhui, Duan Yuting) (2011)

Chai Wenguang proposed [14] an intelligent agent technology with database mining. In this approach there is data acquisition agent which stores data in local database. After that data is send to mining agent for pre processing analysis. The detection for vulnerability is done using alarm evaluate model which create an alarm for every attack detected. The intelligent intrusion detection with web data mining is also compared with traditional intrusion detection system in which it is found more efficient. The weakness of this approach is that there is need to improve data mining algorithms for improving efficiency of proposed mechanism.

K. "CIDT : Detection of malicious code injection attacks on web application" (Atul S. Choudhary and M.L Dhore) (2012)

Atul S. Choudhary proposed [18] code injection detection tool. In this approach there are two detectors Query Detector and Script Detector. Query Detector framework consists of a malicious keyword text file. If user input consist of any malicious keyword query detector will not allow database connectivity. When database connection occurs for non malicious input Script Detector further filters the input. It filters HTML content to prevent SQL and XSS attack. This approach has been implemented in ASP.Net language and experiment shows successful prevention of SQL and XSS attack. The weakness of this approach is that it is not effective against stored procedure attack.

VI. Cross Site Detection And Prevention

A. ¡°BIXAN: Browser Independent XSS Sanitizer For Prevention of XSS attacks " (Sharath Chandra v ,S.Selvekumar) (2011)

It has been proposed in [16] a technique which is invoked when user injects code in the field of web application. The HTML content is passed to the XSS sanitizer. Sanitizer parses the HTML content and checks the presence of static tags. The static tags are retained while rests of tags are filtered out. Even static tags contain dynamic content which are filtered out by JavaScript tester. After filtering HTML content is converted into DOM. The sanitized user content is retained in DOM. This approach was tested by cheat sheet [21] contained 114 scripts and result is obtained in various web browser. It was found that all 114 scripts were filtered out but BIXAN .Moreover it reduces the anomalous behavior of web browsers. The weakness of this solution is that it is at the server side and browser source need to be modified for obtaining results.

B. "S^{2}XS^2: A Server Side Approach to Automatically Detect XSS Attacks" (Hossain Shahriar and Mohammad ZULKernine) (2011)

Hossain Shahriar [17] proposed a detection framework which has six modules. The first module Boundary Injecton injects boundary for content generation location .They apply

DOI: 03.LSCS.2013.3.98

two types of boundaries HTML Comment and JavaScript comment. Comments are injected with token to identify duplicity. The policy is generated for attack detection. The second module is Policy Storage which stores policy for attack detection. The third module is Web Server which represents web program container where injected boundaries programs can be accessed. Web Server generates response pages and forwards it to next module. The fourth module is Feature Comparator which compares response page with policy rules. The fifth module is Attack Detector which detects and removes malicious code and forwards response to next module. The Boundary Remover is last module which removes boundary injected during first module from safe content. This approach was tested in java and results were obtained with zero false negative. It also detects advanced XSS. The weaknesses of this approach are; it suffers from false positive and response delays. The policy checking phase is also exhaustive.

VII. Broken Authentication & Session Management Detection & Prevention

A. ¡°A process for supporting risk-aware web authentication mechanism choice " (Karen Renaud) (2006)

In [6] Karen Reynaud proposes a risk aware procedure for developing web authentication mechanism to control authentication vulnerabilities. The proposed methodology is simple and well structured for enabling developers to a choose web authentication mechanism for the particular website. The steps are defined for choosing authentication mechanism. In step one the target user group is defined for classifying users for particular web application. In next step possible impact of an intrusion is defined i.e. how intrusion can affect web application. In third step estimated budget is decided for features one can afford. In fourth step authentication mechanism is chosen by ranking different mechanism using proposed formula

Opportunity = ((Guessability + Observability +Recordability + Analysability)/Resisitibility)

This formula provides weighting to the authentication mechanism which helps in providing rank to them. In next step mechanism is confirmed and plans for testing are determined. This procedure provides a process for supporting authentication mechanism for web application. The weakness of this approach is that this approach needs developer knowledge for authenticating web application.

B. ¡°Automatic Detection Of Session Fixation Vulnerabilities in Web Applications" (Yusuki Takamastu ,Yuji Kosuga ,Kenji Kono) (2012).

In [9] Yusuke Takamastu proposes a technique to check session fixation attack. Session fixation is an attack in which attacker forces visitor to use session ID passed by him and can use application as a visitor. In this approach they designed a system which works in three steps. First step is Packet Capturing captures all the packets observe the change of sessionIDs. The system lies between the user browser and the web application server. In second step Initial Inspection is done for checking the vulnerability for session ID. The third step is Attack Simulation in which system launches attack simulator. The attack simulator automatically generates the same environment as a real attacker performs as a virtual attacker and a victim. At this time virtual attacker access and login to web application with Session ID (SID) that attacker obtained. The virtual attacker checks whether he can login with the obtained SID or not. The response is checked for keywords. For example "welcome victim" if such type of keyword is obtained then web application is considered as vulnerable. The weakness of this approach is that it is difficult to manage such kind of virtual environment for real world application.

VIII. Future Work

Web application security is extremely important for using and providing better services over internet. The main motive of application developer is to implement the business logic correctly but despite of proper functioning of web application it may be vulnerable to dangerous attack which can harm organization database and reputation. Our future work includes proposal of a web application security tool which can detect and prevent the most of the web application attacks. Our proposed mechanism will mainly detect and prevent the attacks discussed in this paper with ease of implementation and fast detection with zero false positive & negative. We will also propose an approach for secure implementation of web application as the lack of security awareness is main reason for web application attack.

Conclusions

SQL Injection, XSS, Broken authentication and session management are the most dangerous and the common web application attacks. This survey presents study of current techniques against these attacks. The present techniques suffers weaknesses such as

- Complex framework.
- Incomplete implementation.
- Real world implementation issues.
- Response delay.
- Average time in detection.
- Detection of non signature attack.
- False positive and false negative.
- Developer knowledge.

Industry has been developed many promising techniques to secure web application but despite of the effectiveness of these techniques it is necessary for application developer to use them as a part of developing process. With increasing use of web application it is important for software developers to follow appropriate security framework during software development life cycle for ensuring better services to end users.

DOI: 03.LSCS.2013.3.98

REFERENCES

[1] Imperva Web Application Attack Report 2012. http:// www.imperva.com. (Accessed in Dec 2012)

[2] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.(Accesed in Jan 2013)

[3] https://www.owasp.org/index.php/Broken_Authentication_and_Session_Management (Accessed in Jan 2013)

[4] A. McClure and Ingolf H.Kruger, ," SQL DOM : Compile Time Checking of Dynamic SQL Statements" , International Conference of Software Engineering May 2005 , ACM ,PP 88-96.

[5] Xiang Fu ,Xin Lu , "A Static Analysis Framework For Detecting SQL Injection Vulnerabilities" 31st Annual International Computer Software And Application Conference , IEEE, 2007 .

[6] Karen Renaud , "process for supporting risk-aware web authentication mechanism choice " Reliability Engineering and System Safety Elsevier 2007 PP -1217

[7] YongJoon Park , JaeChul Park , "Web Application Intrusion Detection System For Input Validation Attack" , Third 2008 International Conference On Convergence And Hybrid Information Technology ,IEEE, PP 498-504.

[8] Jin-Cherng Lin , Jan-Min Chen , Cheng-Hsiung Liu , "An Automatic Mechanism For Sanitizing Malicious Injection " , The 9th International Conference For Young Computer Scientists 2008 ,IEEE, PP 1470-1475.

[9] Anyi liu ,yi yuan , "SQLProb : A Proxy based Architecture towards preventing SQL injection attacks " , sac'march 2009, ACM ,PP.2054-2061.

[10] Meijunjin ,"An Approach For Sql Injection Vulnerability Detection" , 2009 Sixth International Conference On Information Technology :New Generations IEEE , PP 1411-1414.

[11] Abdul Razzaq ,Ali Hur , Nasir Haider , "Multi Layer Defense Against Web Application " , 2009 Sixth International Conference On Information Technology :New Generations , IEEE ,PP.492-497

[12] Yang Haixia And Nan Zhihong , "A Database Security Testing Scheme Of Web Application" , 4th International Conference On Computer Science And Education,2009 , IEEE, PP .953-955.

[13] Jeom-Goo Kim , "Injection Attack Detection Using Removal Of Sql Query Attribute Values IEEE 2011.

[14] Chai Wenguuang, Tan Chunhui, Duan Yuting, " Research Of Intelligent Intrusion Detection System Based On Web Data Mining Technology", IEEE 4th International Conference On Business Intelligence And Financial Engg. 2011, PP. 14-17.

[15] Nilesh Kochre , Satish Chalukar ,,Santosh Kakde , "Survey On SQL Injection Attacks And Their Countermeasures " , International Journal Of Computational Engineering And Management , Vol -14 , October 2011 .

[16] Sharath Chandra V. , S.Selvekumar , "BIXAN:Browser Independent XSS Sanitizer For Prevention Of XSS Attacks.ACM SIGSOFT ,September 2011 Volume 36 Number 5.

[17] Hossain Shaihriar And Mahammad Zulkernine , "S^2XS^2 :A Server Side Apptoch To Automatically Detect XSS Attacks , 2011 IEEE Ninth International Conference On Dependable ,Automatic Secure Computing , PP.7-17

[18]] Atul S. Choudhary And M.L Dhore , "CIDT : Detection Of Malicious Code Injection Attacks On Web Application" , International Journal Of Computing Applications Volume-52-N0.2, August 2012 , PP. 19-25.

[19] Yusuki Takamastu ,Yuji Kosuga ,Kenji Kono "Automatic Detection Of Session Fixation Vulnerabilities "2012 Tenth Annual International Conference on Privacy, Security and TrustIEE ,PP-112-119.

[20] Rahul Johri , Pankaj Sharma , A Survey On Web Application Vulnerability Exploitation And Security Engine For SQL Injection , 2012 International Conference On Communication System And Network Technologies , IEEE , PP.453-458

[21] XSS prevention cheat sheet OWASP https://www.owasp.org/index.php/XSS_ (Cross_Site_Scripting)_ Prevention_ Cheat_Sheet.

DOI: 03.LSCS.2013.3.98